CS 61A, Summer 2006 July 17, 2006
LAB ASSIGNMENT 4A

Start by reading SICP section 2.3.3 (pages 151–161).

1. SICP ex. 2.62.

2. The file `~cs61a/lib/bst.scm` contains the binary search tree procedures from pages 156–157 of SICP. Using `adjoin-set`, construct the trees shown on page 156.

3. SICP ex. 2.74.

4. The file `~cs61a/lib/scheme1.scm` contains the scheme-1 interpreter. To start the interpreter, type `(scheme-1)`. Familiarize yourself with it by evaluating some expressions. Remember: you have all the Scheme primitives for arithmetic and list manipulation; you have `lambda` but not higher-order functions; you dont have `define`. To stop the scheme-1 interpreter and return to STk, just evaluate an illegal expression, such as `()`.

4a. Trace in detail how a simple procedure call such as `((lambda (x) (+ x 3)) 5)` is handled in scheme-1. You may want to trace the procedure `substitute` thats part of the interpreter.

4b. Since all the Scheme primitives are automatically available in scheme-1, you might think you could use STks primitive `map` function. Try these examples:

```
Scheme-1: (map first (the rain in spain))
Scheme-1: (map (lambda (x) (first x)) (the rain in spain))
```

Explain the results.

4c. Modify the interpreter to add the `and` special form. Test your work. Be sure that as soon as a false value is computed, your `and` returns `#f` without evaluating any further arguments.